(19) Europäisches Patentamt
European Patent Office
Office européen des brevets

(11) EP 1 130 877 A2

(12) EUROPEAN PATENT APPLICATION

(43) Date of publication:
05.09.2001 Bulletin 2001/36

(51) Int Cl.[7]: **H04L 29/06**

(21) Application number: **01440031.1**

(22) Date of filing: **14.02.2001**

(84) Designated Contracting States:
AT BE CH CY DE DK ES FI FR GB GR IE IT LI LU MC NL PT SE TR
Designated Extension States:
AL LT LV MK RO SI

(30) Priority: **02.03.2000 US 517168**

(71) Applicant: **ALCATEL**
75008 Paris (FR)

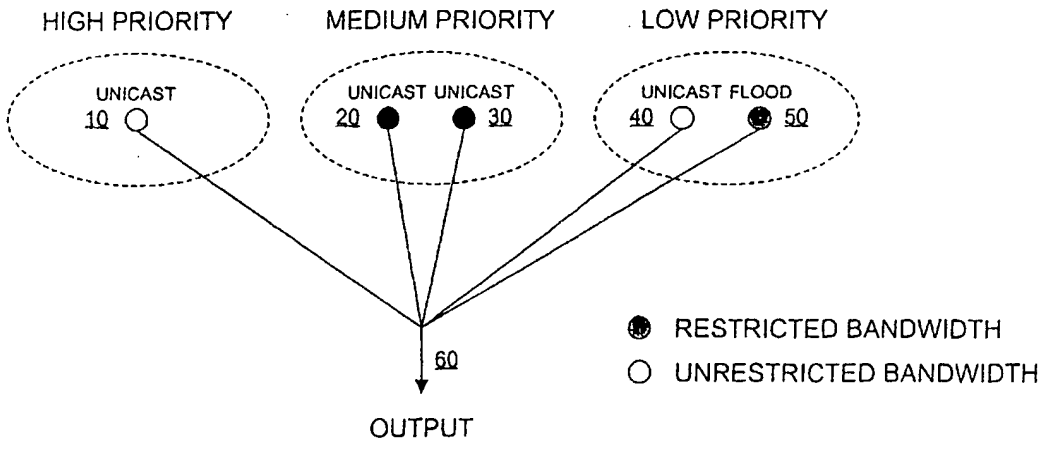
(72) Inventor: **Bretagna, Drew**
San Jose, California 95123 (US)

(74) Representative: **Schäfer, Wolfgang, Dipl.-Ing.**
Dreiss, Fuhlendorf, Steimle & Becker
Postfach 10 37 62
70032 Stuttgart (DE)

(54) Qualified priority queue scheduler

(57) A queue scheduling method and apparatus for a queuing structure including a plurality of queues at different priorities and having different bandwidth privileges competing for the bandwidth of an output. The method selects a queue at the current priority, having data for release to the output and having available bandwidth to release data to the output. If there are a plurality of such queues, round-robin ordering determines which queue in the plurality is selected. The apparatus includes a plurality of bit masks representing the status in relation to a queue state variable of the different queues, which masks are combined in a bit-wise AND operation to determine which queues are selectable for releasing data to the output.

FIGURE 1



EP 1 130 877 A2

Printed by Jouve, 75001 PARIS (FR)

Description

BACKGROUND OF THE INVENTION

[0001] Data communication switches receive packets on a plurality of inputs and transmit them on a plurality of outputs. Sometimes, packets come in on two or more inputs destined for the same output. To minimize collisions between such packets, queueing is used. Queues temporarily store some packets while others are consuming the bandwidth necessary for delivery of the queued packets to an output.

[0002] The release of packets from queues competing for output bandwidth is typically coordinated by a scheduler. Many schedulers honor a strict rule of priority scheduling, that is, queues are assigned different priority levels and those associated with relatively high priority levels are allowed to release packets before those associated with relatively low priority levels. Strict priority scheduling allows flows which have been designated as relatively time-critical to take precedence over those which have been designated as relatively less time critical.

[0003] Despite its general efficacy, strict priority scheduling is not problem-free. One undesirable effect is priority blocking. Priority blocking occurs when delivery of packets from relatively low priority queues suffer a sustained delay due to servicing of higher priority queues. Priority blocking can eventually cause packets in the blocked queues to be overwritten while awaiting release, a condition known as dropping. Dropping undesirably disrupts the flows to which the dropped packets relate.

[0004] While priority scheduling is by design supposed to provide faster service to some queues than others, departure from a strict rule of prioritization may be warranted in certain cases. For instance, priority blocking may be caused by "stuffing" of a single queue at the priority level receiving service with non-critical or even redundant (e.g., in the case of an unresolved spanning tree loop) traffic. Strict adherence to priority scheduling would allow this "stuffing" to in effect hold all of the output bandwidth hostage at the expense of the relatively low priority queues.

[0005] Accordingly, there is a need for a qualified priority scheduling method for reducing priority blocking through the imposition of reasonable limitations on a general policy of prioritization. There is also a need for efficient scheduling hardware to effectuate the qualified priority scheduling.

SUMMARY OF THE INVENTION

[0006] The present invention provides a qualified priority scheduling method and apparatus for a data queuing structure including a plurality of groups of queues competing for the bandwidth of an output. Each group has at least one queue and is associated with a different priority level. Moreover, at least one queue within at least one group is a restricted bandwidth queue, i.e. has a bandwidth contract which cannot be exceeded.

[0007] In accordance with the queue scheduling method, a queue within the group associated with the current priority level which (i) has data for release to the output (ii) has available bandwidth, i.e. has not violated its bandwidth contract (if any), is selected to release data to the output. If there are a plurality of such queues, round-robin ordering determines which queue in the plurality is selected. The current priority level is initially set at the highest priority level and is decremented until a queue is selected or all priority levels have been checked, whichever occurs first. Queues may have restricted or unrestricted bandwidth. Unrestricted bandwidth queues always have available bandwidth, whereas restricted queues have available bandwidth only if they have credit. Credit is allocated to restricted queues periodically and is reduced in accordance with the length of released data. Data are released in quanta of credit.

[0008] In accordance with the queue scheduling apparatus, a selector is arranged for selecting a queue from the plurality of queues as a function of a plurality of queue state variables and transmitting queue selection information. The selector preferably includes a plurality of bit masks representing the status in relation to a queue state variable of the different queues, which masks are combined in a bit-wise AND operation to determine which queues are selectable for releasing data to the output. The selector preferably further includes an arbiter for receiving the result of the determination, for selecting a single queue round-robin from the selectable queues and for transmitting the result of the round-robin selection to a driver for releasing from the selected queue.

[0009] These and other aspects of the present invention may be better understood by reference to the following detailed description taken in conjunction with the accompanying drawings briefly described below.

BRIEF DESCRIPTION OF THE DRAWINGS

[0010]

Figure 1 is a policy level diagram of an exemplary shared-output queuing structure with scheduling characteristics;

Figure 2 is a block diagram of a preferred scheduling apparatus for a shared-output queuing structure and an associated queuing structure;

Figure 3 is a block diagram of the multi-stage selector of Figure 2;

Figure 4 is a block diagram of the round-robin arbiter of Figure 2;

Figure 5 is a block diagram of statistics of Figure 2;

Figure 6 is a flow diagram of priority-bandwidth queue selection steps;

Figure 7 is a flow diagram of qualifying queue selection steps of round-robin arbitration; and
Figure 8 is a flow diagram of winning queue selection steps of round-robin arbitration.

DETAILED DESCRIPTION OF THE PREFERRED EMBODIMENT

[0011] The present invention is primarily directed to a method and apparatus for scheduling the release of data from a plurality of queues competing for the bandwidth of an output. In Figure 1, an exemplary shared-output queuing structure with scheduling characteristics is shown at a policy level. In the exemplary structure, there are five queues 10, 20, 30, 40, 50 competing for the bandwidth of output 60. The release of data from queues 10, 20, 30, 40, 50 is scheduled, i.e. time-multiplexed, to avoid contention. Queues 10, 20, 30, 40, 50 are characterized by priority level, including high priority queue 10, medium priority queues 20, 30 and low priority queues 40, 50 for releasing data to output 60. Queues 10, 20, 30, 40, 50 are further characterized by bandwidth type for releasing data to output 60. The sole high priority queue 10 is an unrestricted bandwidth queue. Both medium priority queues 20, 30 are restricted bandwidth queues. One low priority queue 40 is an unrestricted bandwidth queue, whereas the other low priority queue 50 is a restricted bandwidth queue. Queues 10, 20, 30, 40, 50 are further characterized by flow type, i.e. unicast or flood. However, flow type has no independent bearing on the order in which data are released to output 60. Of course, the shared-output queuing structure illustrated in Figure 1 is merely one example of such structure; the number, priority level and bandwidth type of queues will differ in other shared-output queuing structures operative in accordance with the present invention depending on the policies it is desired to implement.

[0012] In a preferred scheduling method for a shared-output queuing structure, a queue within the group associated with the current priority level which has data for release to the output and has available bandwidth, i.e. has not violated its bandwidth contract, if any, is selected to release data to the output. If there are a plurality of such queues, round-robin ordering resolves which queue in the plurality is selected. Various elaborations of this basic scheduling method are possible, as described hereinafter. Nevertheless, at a fundamental level, this basic method, despite its apparent simplicity, is believed to confer a significant advance over the scheduling methods of the prior art.

[0013] Applying the preferred scheduling method to the exemplary shared-output queuing structure illustrated in Figure 1, if unrestricted bandwidth high priority queue 10 has data for release to output 60, queue 10 is selected. If queue 10 is not selected, restricted bandwidth medium priority queues 20, 30 are checked. More particularly, if queues 20, 30 each have data for release to output 60, and each have available bandwidth, the next-in-line one of queues 20, 30 in round-robin order is selected. If queues 20, 30 each have data for release, but only one of queues 20, 30 has available bandwidth, the one of queues 20, 30 with available bandwidth is selected. If only one of queues 20, 30 has data for release and available bandwidth, that one of queues 20, 30 is selected. If, neither of queues 20, 30 is selected, low priority queues 40, 50 are checked. More particularly, if queues 40, 50 each have data for release to output 60, and restricted bandwidth low priority queue 50 has available bandwidth, the next-in-line one of queues 40, 50 in round-robin order is selected. If queues 40, 50 each have data for release, but restricted bandwidth low priority queue 50 does not have available bandwidth, unrestricted bandwidth low priority queue 40 is selected. If only queue 40 has data for release, queue 40 is selected. If only queue 50 has data for release, queue 50 is selected if queue 50 has available bandwidth. If neither of queues 40, 50 is selected, no data are released to output 60.

[0014] Referring now to Figure 2, a preferred shared-output scheduling apparatus is shown at a component level along with an associated queue structure. Queuing structure 210 includes data queues 0 through N for receiving data on respective ones of inputs 0 through N 212 and for releasing data to output 214. Release of data from queuing structure 210 is controlled by scheduler 200. Scheduler 200 includes manager 220, multi-stage selector 230, driver 240 and statistics 250. Manager 220 retrieves data from statistics 250 and updates selector 230 and statistics 250. Selector 230 includes selection masks 232, priority-bandwidth selector 234 and round-robin arbiter 236 for facilitating queue selection. Selector 230 selects data queues for releasing data to output 214 and transmits queue selection information to driver 240. Driver 240 receives queue selection information, retrieves data from statistics 250, updates statistics 250, and controls release of data from the selected queues.

[0015] Referring now to Figure 3, selector 230 is shown in greater detail. Selector 230 has selection masks 232 including enablement mask 312, backlog mask 314 bandwidth mask 316 and priority masks 318. Each mask maintains a status bit for each of queues 0 through N in memory element 210. Enablement mask 312 indicates which among queues 0 through N are enabled for transmitting data to output 214. A "1" in the bit position reserved for the queue represents enablement, whereas a "0" represents non-enablement. Backlog mask 314 indicates which among queues 0 through N have data awaiting transmission to output 214. A "1" in the bit position reserved for the queue represents the presence of pending data, whereas a "0" represents the absence of pending data. Bandwidth mask 316 indicates which among queues 0 through N has bandwidth for transmitting data to output 214. A "1" represents the availability of bandwidth, whereas a "0" represents the unavailability of bandwidth. Priority masks 318 collectively indicate the priority level of queues 0 through N.

A separate priority mask is maintained for each priority level. A "1" in the bit position reserved for the queue in a priority mask indicates assignment of the queue to the priority level which the mask represents, whereas a "0" indicates non-assignment to the priority level. Masks 232 are configurable by manager 220. Backlog mask 314 and bandwidth mask 316 are updated by manager 220 in response to information retrieved from statistics 250.

[0016] Selection masks 232 are coupled to priority-bandwidth selector 234. Selector 234 facilitates queue selection by reducing the selection to a round-robin selection among participating queues. When participant selection is enabled, enablement mask 312, backlog mask 314, bandwidth mask 316 and one of priority masks 318 at a time are submitted to mask compare 324 for a bit-wise AND operation. The one of priority masks 318 submitted to the bit-wise AND operation is determined by priority counter 328, which instructs multiplexor 322 to release one of priority masks 318 based on the current value of counter 328. For each participant selection round, counter 328 selects the highest priority mask first and selects priority masks of incrementally lower priority until winner select enablement is indicated at the current priority level, or all priority masks have been submitted.

[0017] The result of the bit-wise AND operation performed by mask compare 324 is a mask indicating which queues at the current priority level, if any, are entitled to participate in winner selection performed in round-robin arbiter 236. Particularly, the resultant mask has a "1" at all bit positions reserved for queues which are enabled for output port 214, have pending data, have available bandwidth and are at the current priority level, if any, and has a "0" at other bit positions. Thus, the resultant mask indicates any participating queues by a "1" and any non-participating queues by a "0". The resultant mask is submitted to winner select enable 326 for an OR operation. If at least one bit position in the resultant mask has a "1", the OR operation yields a "1" and winner selection is enabled at the current priority level. Otherwise, the OR operation results in a "0", winner selection is not enabled at the current priority level and the bit-wise AND operation is performed at the next priority level, if any. The OR operation result is supplied as feedback to counter 328 to notify counter 328 whether an additional priority mask selection is required. It will be appreciated that if a "null" mask results from the bit-wise AND operation at all priority levels, winner selection is not enabled and the round-robin arbitration is not conducted.

[0018] Priority-bandwidth selector 234 is coupled to round-robin arbiter 236. Round-robin arbiter 236 resolves a winning queue by making a round-robin selection among participating queues as determined by selector 234. Turning to Figure 4, round-robin arbiter 236 is shown in greater detail. When winner selection is enabled, the participating queue mask, i.e. the mask resulting from the bit-wise AND operation performed in selector 234, is submitted to preliminary select array 410 for qualifying queue determinations. Particularly, each preliminary select element in array 410 receives a subset of bits from the participating queue mask and selects a single qualifying queue from among the participating queues, if any, that it represents. More particularly, the preliminary select element whose qualifying queue was selected as the winning queue by final select 420 in the last round-robin arbitration selects as a qualifying queue the next one of its participating queues in round-robin order, if any. The other preliminary select elements select as a qualifying queue the first one of their respective participating queues, if any. The qualifying queue selections are submitted to final select 420 along with any "no qualifier" notices from any preliminary select elements not representing any participating queues. Moreover, if the preliminary select element whose qualifying queue was selected as the last winning queue wrapped-around, i.e. returned from its last to first queue in the round-robin order in the course of making the selection, the preliminary select element submits a "wrap-around" notice to final select 420. Final select 420 selects as a winning queue the qualifying queue from the preliminary select element whose qualifying queue was selected as the last winning queue, unless that preliminary select element submitted a "no qualifier" or "wrap-around" notice. If that preliminary element submitted a "no qualifier" or "wrap-around" notice, final select 420 selects as the winning queue the qualifying queue from the next one of the preliminary select elements in round-robin order, if any. Final select 420 transmits the winning queue identifier to queue driver 240 in completion of the round-robin arbitration. The winning queue identifier is also supplied as feedback to array 410 for use by preliminary select elements in subsequent qualifying queue selections.

[0019] Referring now to Figure 2 in conjunction with Figure 5, in response to the winning queue identifier received from selector 230, driver 240 consults statistics 250 to retrieve queue state information and controls the release of data from the winning queue to output 214. Statistics 250 maintain multi-field entries for each of queues 0 through N within queuing structure 210 including a queue identifier, a head pointer, a tail pointer, current depth, maximum depth, current credit and total credit values. The winning queue identifier is used to "look-up" the corresponding entry in statistics 250. A predetermined quantum of credit is added to the current credit value for the corresponding entry. A quantum may be advantageously selected in relation to the maximum length of a packet formatted in accordance with a particular protocol, such as 1518 bytes for an Ethernet packet. Driver 240 controls release of data from the winning queue to output 214 beginning with the data at the queue location indicated by the head pointer in the corresponding entry, until all data have been released from the winning queue, the current credit has been exhausted or, in the case of a restricted bandwidth queue, or the

total credit has been exhausted, whichever occurs first. Driver 240 updates the head pointer, current depth, total credit value and current credit values for the corresponding entry as data are released to reflect the changing queue state. The current credit value is "zeroed out" if all data have been released or, in the case of a restricted bandwidth queue, the total credit is exhausted before the current credit.

[0020] Manager 220 consults statistics 250 to ensure that selection masks 232 reflect the queue state changes made by driver 240. Current depth values in entries are consulted to refresh backlog mask 312 and total credit values in entries are consulted to refresh bandwidth mask 316. Manager 220 also periodically refreshes total credit values in entries maintained for restricted bandwidth queues.

[0021] Turning now to Figure 6, a flow diagram of priority-bandwidth queue selection is shown. Participant selection is enabled (610) and the highest priority is selected as the current priority (620). Participating queues at the current priority, if any, are determined based on queue enablement, backlog and bandwidth states and priority (630). If there is at least one participating queue at the current priority (640), winner selection is enabled and the round-robin arbiter is notified (650). If not, the next highest priority, if any, is selected as the current priority (660). If not all priorities have been checked, the flow returns to Step 630; otherwise, the flow is exited (670).

[0022] Turning to Figure 7, a flow diagram of qualifying queue selection in round-robin arbitration is shown for a representative preliminary select element. The element receives information for the queues the element represents (710) and checks whether the last winning queue was the element's last qualifying queue (720). If the last winning queue is the element's last qualifying queue, the element selects the next (participating or non-participating) queue the element represents in round-robin order (730). If the last winning queue is not the element's last qualifying queue, the element selects the first (participating or non-participating) queue the element represents as the current queue (740). In either event, the element checks whether the current queue has already been checked for participation in this selection round (750). If the current queue has already been checked for participation, the element has no qualifying queue in this selection round and the final selector is notified (755). If the current queue has not already been checked for participation, it is checked (760). If the current queue is a participating queue, the current queue is the qualifying queue and the final selector is notified (765). If the current queue is not a participating queue, a further check is made to determine if the current queue is the last queue the element represents (770). If the current queue is the last represented queue, the element wraps-around, the final selector is notified and the flow returns to Step 740. If the current queue is not the last represented queue, the flow returns to Step 730.

[0023] Turning finally to Figure 8, a flow diagram winning queue selection in round-robin arbitration is shown. The qualifying queues, "no qualifier" notices, if any, and "wrap-around" notices, if any, are received (810) and the preliminary select whose qualifying queue won the last arbitration is selected as the current element (820). A check is made whether the current element has a qualifying queue in this selection round (830). If the current element has a qualifying queue in this selection round, a further check is made whether the current element wrapped-around (840). If the current element did not wrap-around, the qualifying queue from the current element is the winning queue, the queue driver and elements are notified and the flow is exited (850). If, however, the current element does not have a qualifying queue, or the current element wrapped-around, the next element in round-robin order is selected as the current element (860). If the new current element has already been checked in this selection round (870), the previous element is reverted to as the current element and the flow goes to Step 850. If the new current element has not already been checked in this selection round, the flow returns to Step 830.

[0024] It will be appreciated by those of ordinary skill in the art that the invention can be embodied in other specific forms without departing from the spirit or essential character hereof. The present description is therefore considered in all respects illustrative and not restrictive. The scope of the invention is indicated by the appended claims, and all changes that come within the meaning and range of equilvalents thereof are intended to be embraced therein.

Claims

1. A queue scheduling method for a queuing structure having a plurality of queues for releasing data to an output, including a group of one or more queues at a first priority and a group of one or more queues at a second priority, wherein the first and second priorities are different, and a queue having restricted bandwidth, comprising:

   checking the group of one or more queues at the first priority for a queue having data for release and available bandwidth; and
   if one or more queues having data for release and available bandwidth is found at the first priority, releasing data to the output from one of the found queues.

2. The queue scheduling method according to claim 1, further comprising the step of:

   if no queue having data for release and available bandwidth is found at the first priority, checking the group of queues at the second priority for a queue having data for release and available band-

width.

3. The queue scheduling method according to claim 1, further comprising the step of:

if a plurality of queues having data for release and available bandwidth are found at the first priority, selecting round-robin one of the found queues for releasing data to the output.

4. The queue scheduling method according to claim 1, further comprising the step of decreasing the available bandwidth for the queue from which data are released to the output if the queue from which data are released to the output has restricted bandwidth.

5. The queue scheduling method according to claim 1, further comprising the step of increasing credit for the queue from which data are released to the output.

6. The queue scheduling method according to claim 5, further comprising the step of decreasing credit for the queue from which data are released to the output in accordance with the length of the data released.

7. The queue scheduling method according to claim 5, further comprising the step of periodically increasing the available bandwidth of a queue having restricted bandwidth.

8. A queue selector comprising a plurality of masks, each mask having a plurality of bits, each bit representing the status in relation to a queue state variable of a different queue within a plurality of queues coupled to an output, wherein the masks are combined in a bit-wise AND operation to determine which queues within the plurality, if any, are selectable for releasing data to the output.

9. The queue selector according to claim 8, wherein the queue state variables include backlog.

10. The queue selector according to claim 8, wherein the queue state variables include bandwidth availability.

11. The queue selector according to claim 8, wherein the queue state variables include priority.

12. The queue selector according to claim 8, further comprising an arbiter for selecting a queue for releasing data to the output from among the selectable queues.

13. The queue selector according to claim 12, wherein the selection is made round-robin.

14. A scheduling apparatus for a queue structure having a plurality of queues for receiving data on respective ones of inputs and for releasing data to an output, comprising:

a selector for selecting a queue from the plurality of queues as a function of a plurality of queue state variables and transmitting queue selection information; and

a driver for receiving the queue selection information and controlling release of data from the selected queue.

15. The scheduling apparatus according to claim 14, wherein the queue state variables include backlog.

16. The scheduling apparatus according to claim 14, wherein the queue state variables include bandwidth availability.

17. The scheduling apparatus according to claim 14, wherein the queue state variables include priority.

18. The scheduling apparatus according to claim 14, further comprising:

a manager for updating the status of the plurality of queues with respect to one or more of the queue state variables.

FIGURE 1

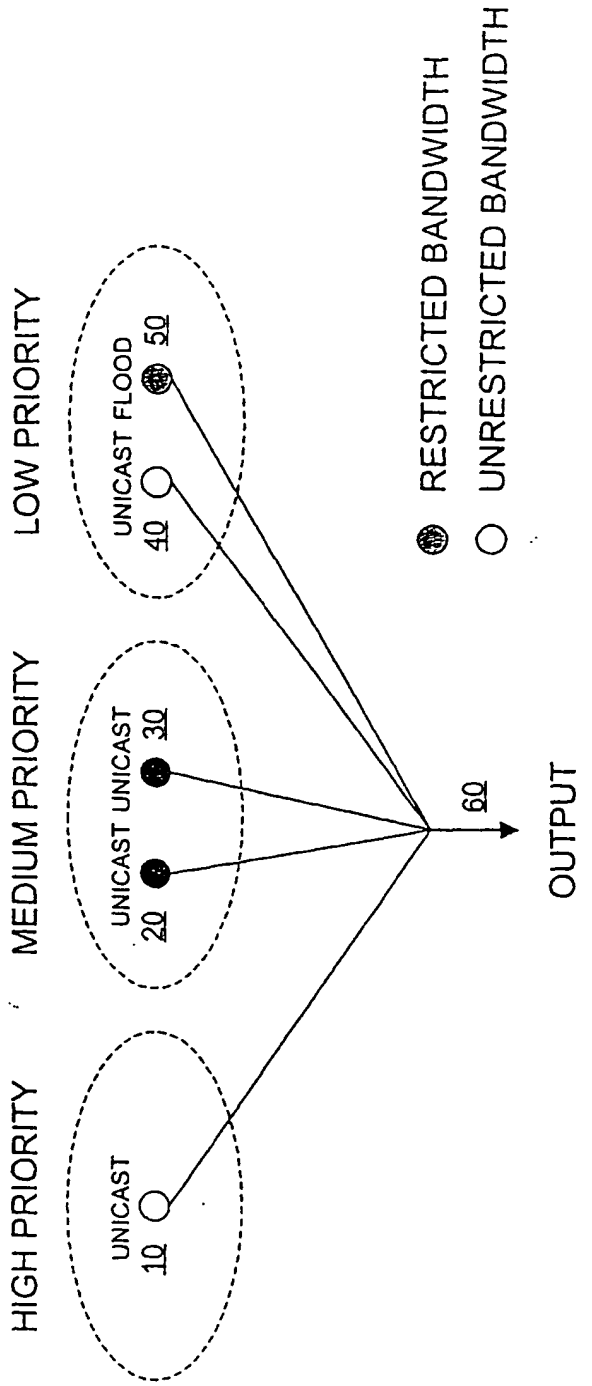
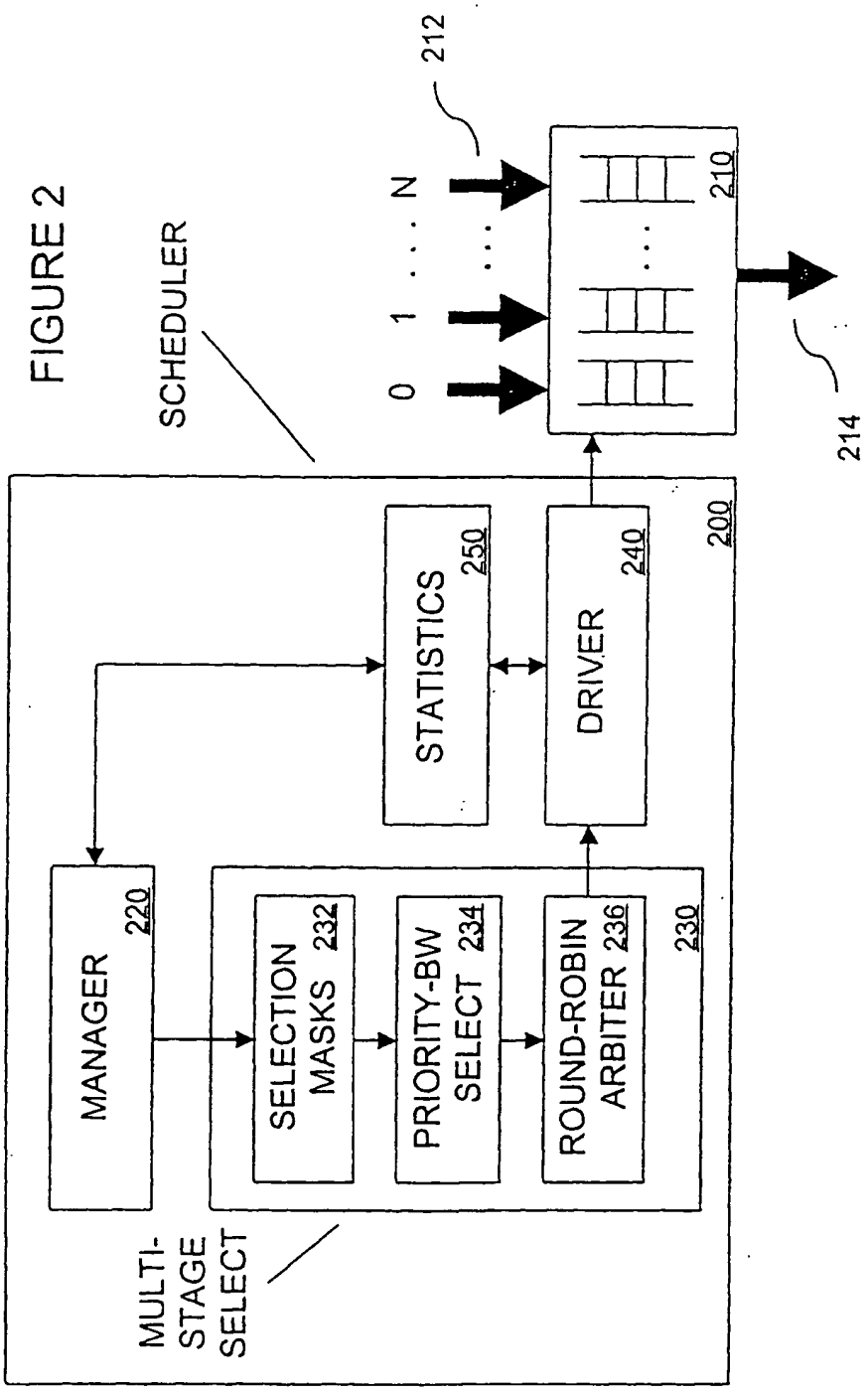

HIGH PRIORITY
MEDIUM PRIORITY
LOW PRIORITY
UNICAST
10
UNICAST UNICAST
20
30
UNICAST FLOOD
40
50
60
OUTPUT
RESTRICTED BANDWIDTH
UNRESTRICTED BANDWIDTH

FIGURE 2


SCHEDULER
0 1 . . . N
212
210
214
200
STATISTICS 250
DRIVER 240
MANAGER 220
SELECTION MASKS 232
PRIORITY-BW SELECT 234
ROUND-ROBIN ARBITER 236
230
MULTI-STAGE SELECT

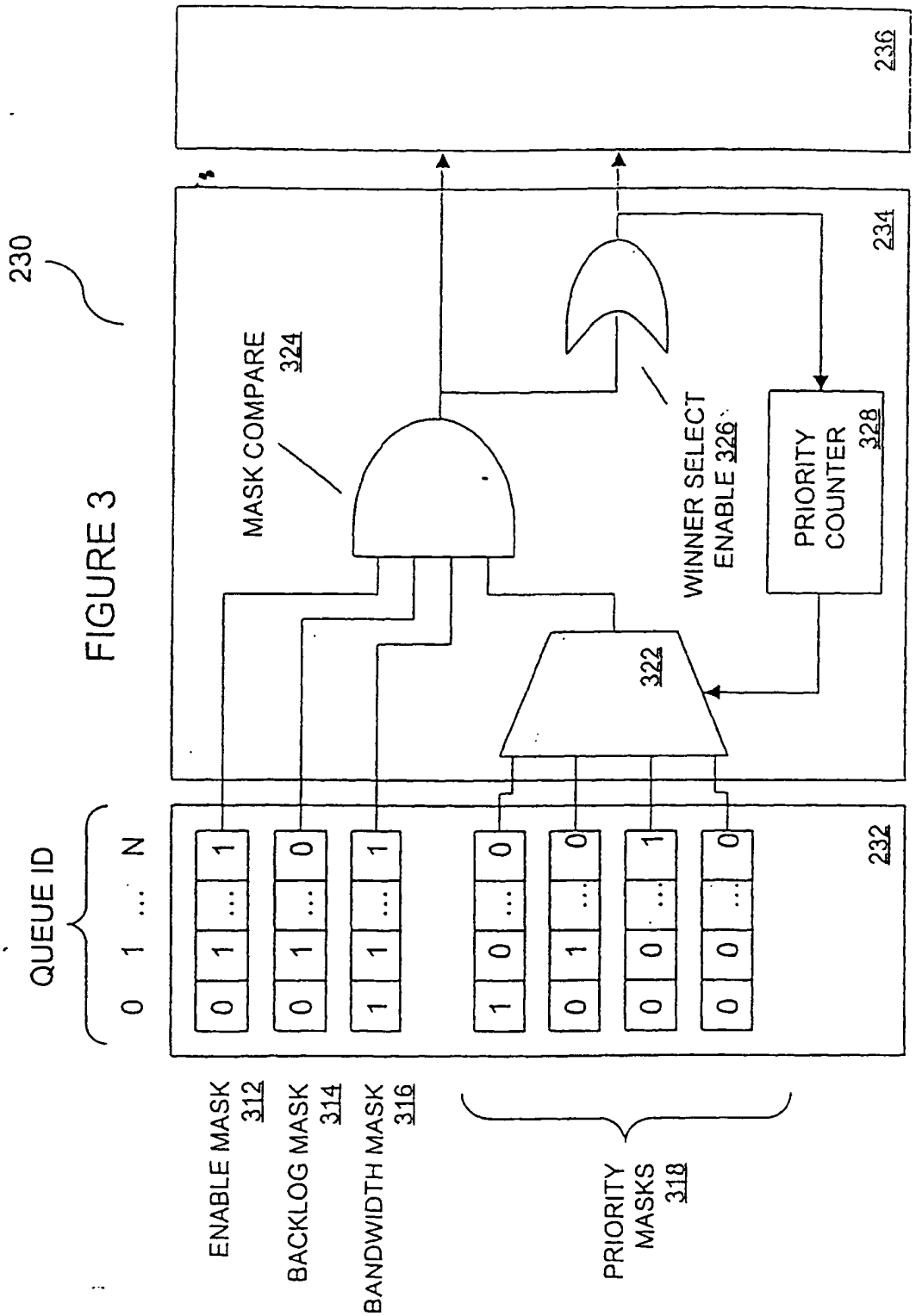

230
236
234
MASK COMPARE
324
WINNER SELECT
ENABLE 326
PRIORITY
COUNTER
328
322
232
QUEUE ID
0 1 ... N
ENABLE MASK
312
BACKLOG MASK
314
BANDWIDTH MASK
316
PRIORITY
MASKS
318

FIGURE 3

FIGURE 4

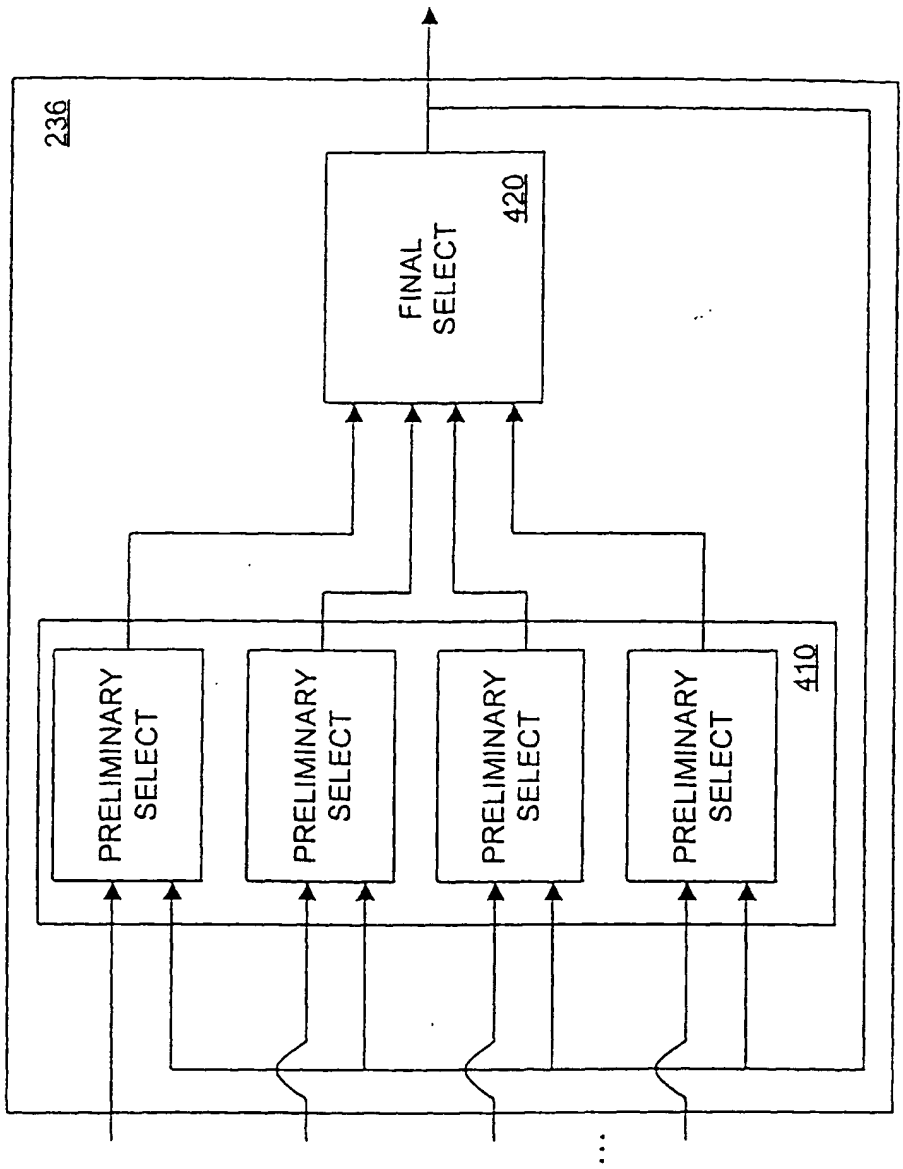
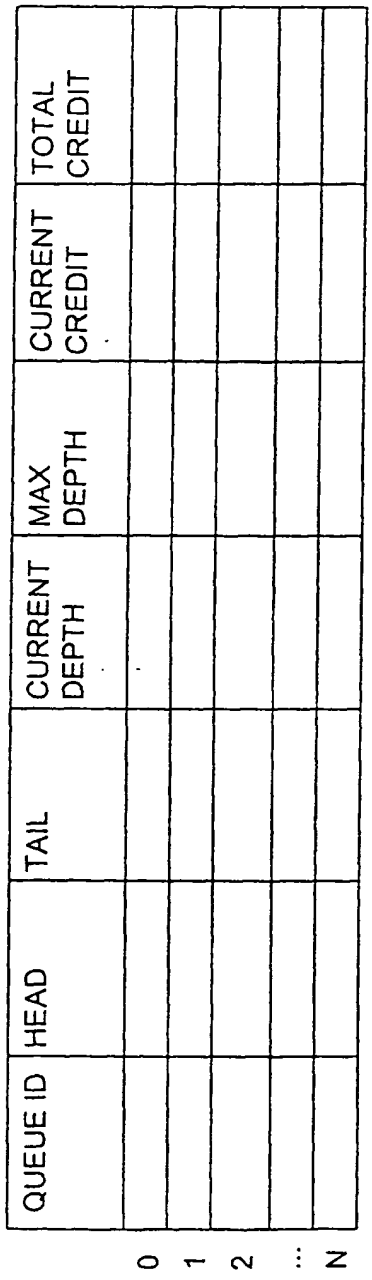

236
FINAL SELECT
420
PRELIMINARY SELECT
PRELIMINARY SELECT
PRELIMINARY SELECT
PRELIMINARY SELECT
410

FIGURE 5


250
QUEUE ID
HEAD
TAIL
CURRENT DEPTH
MAX DEPTH
CURRENT CREDIT
TOTAL CREDIT
0
1
2
⋮
N

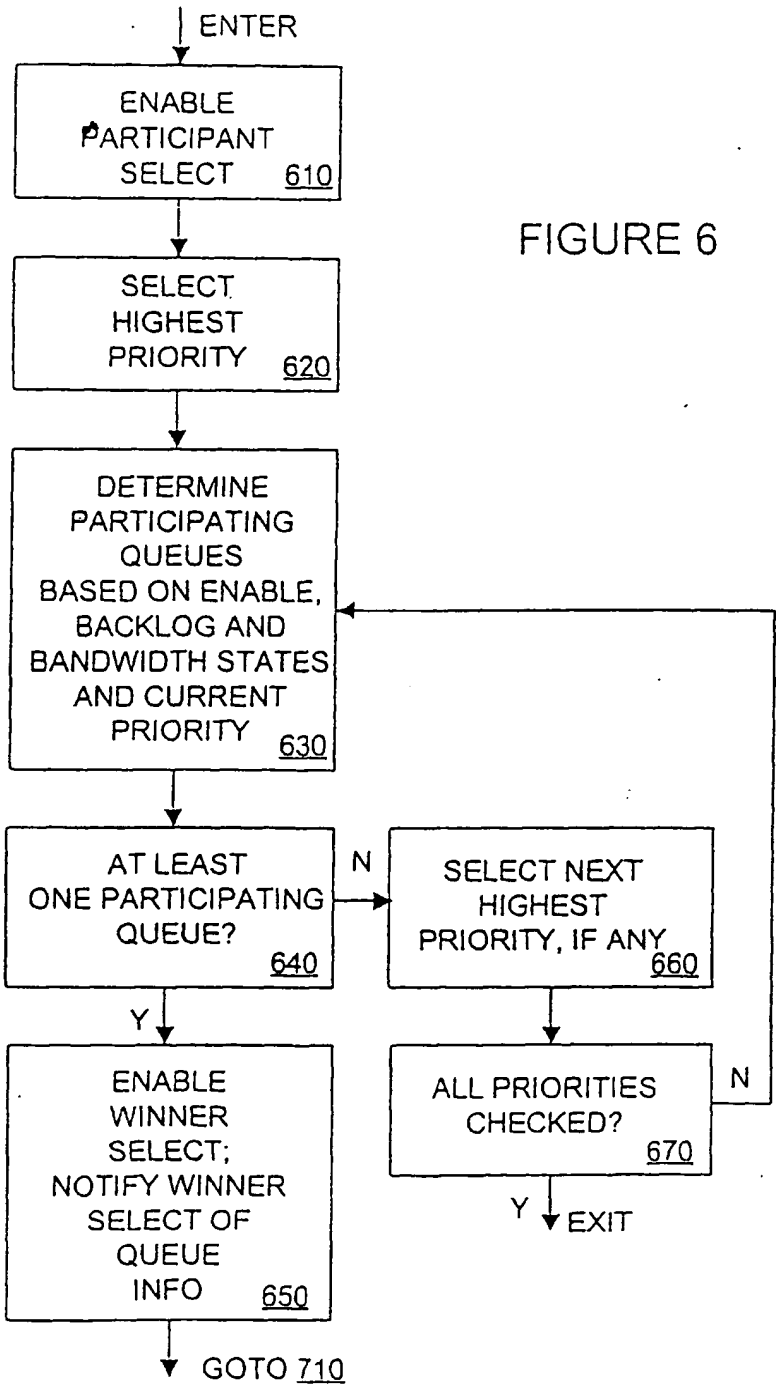

ENTER
ENABLE PARTICIPANT SELECT 610
SELECT HIGHEST PRIORITY 620
DETERMINE PARTICIPATING QUEUES BASED ON ENABLE, BACKLOG AND BANDWIDTH STATES AND CURRENT PRIORITY 630
AT LEAST ONE PARTICIPATING QUEUE? 640
N
SELECT NEXT HIGHEST PRIORITY, IF ANY 660
Y
ENABLE WINNER SELECT; NOTIFY WINNER SELECT OF QUEUE INFO 650
ALL PRIORITIES CHECKED? 670
N
Y
EXIT
GOTO 710

FIGURE 6

FIGURE 7

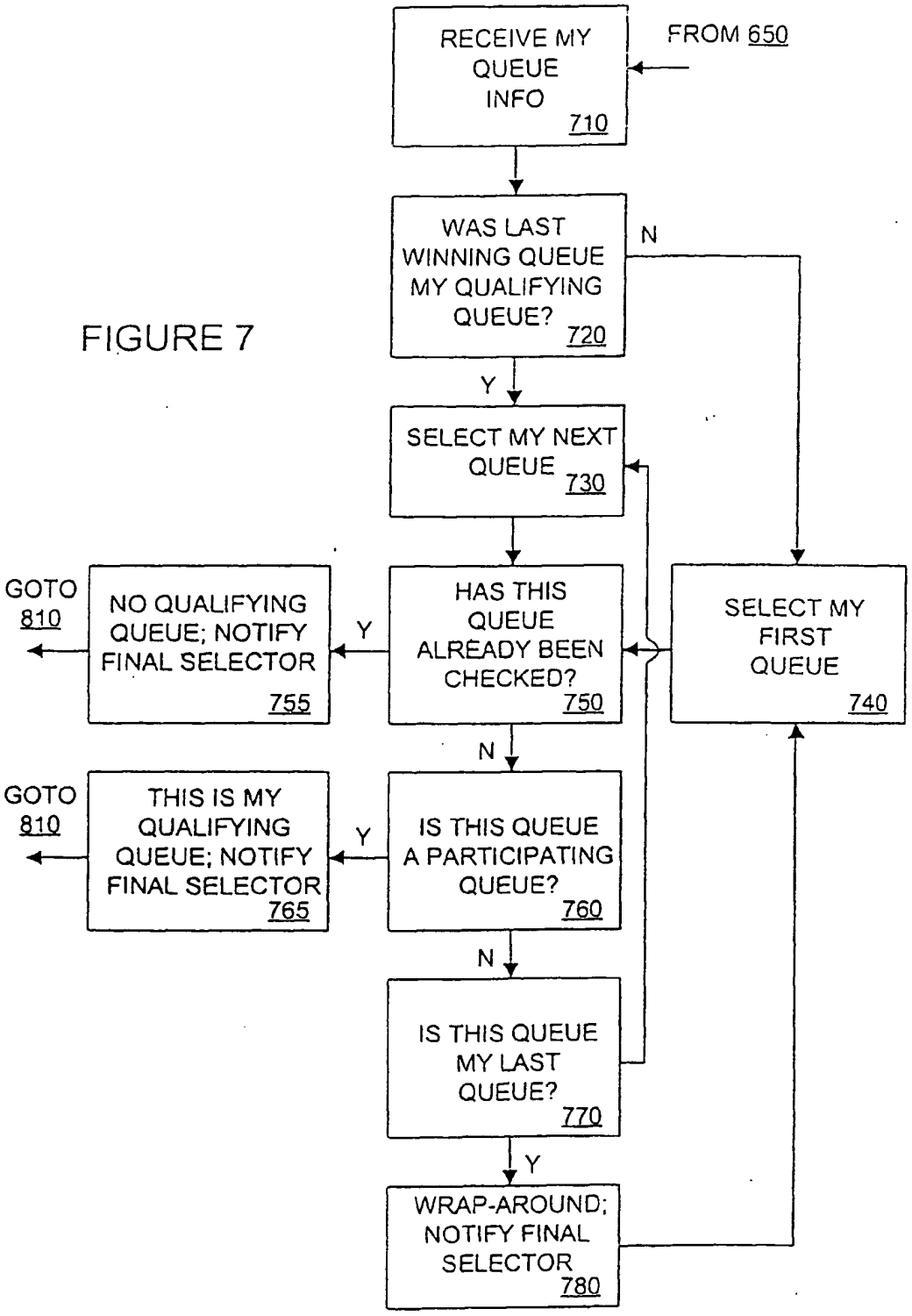
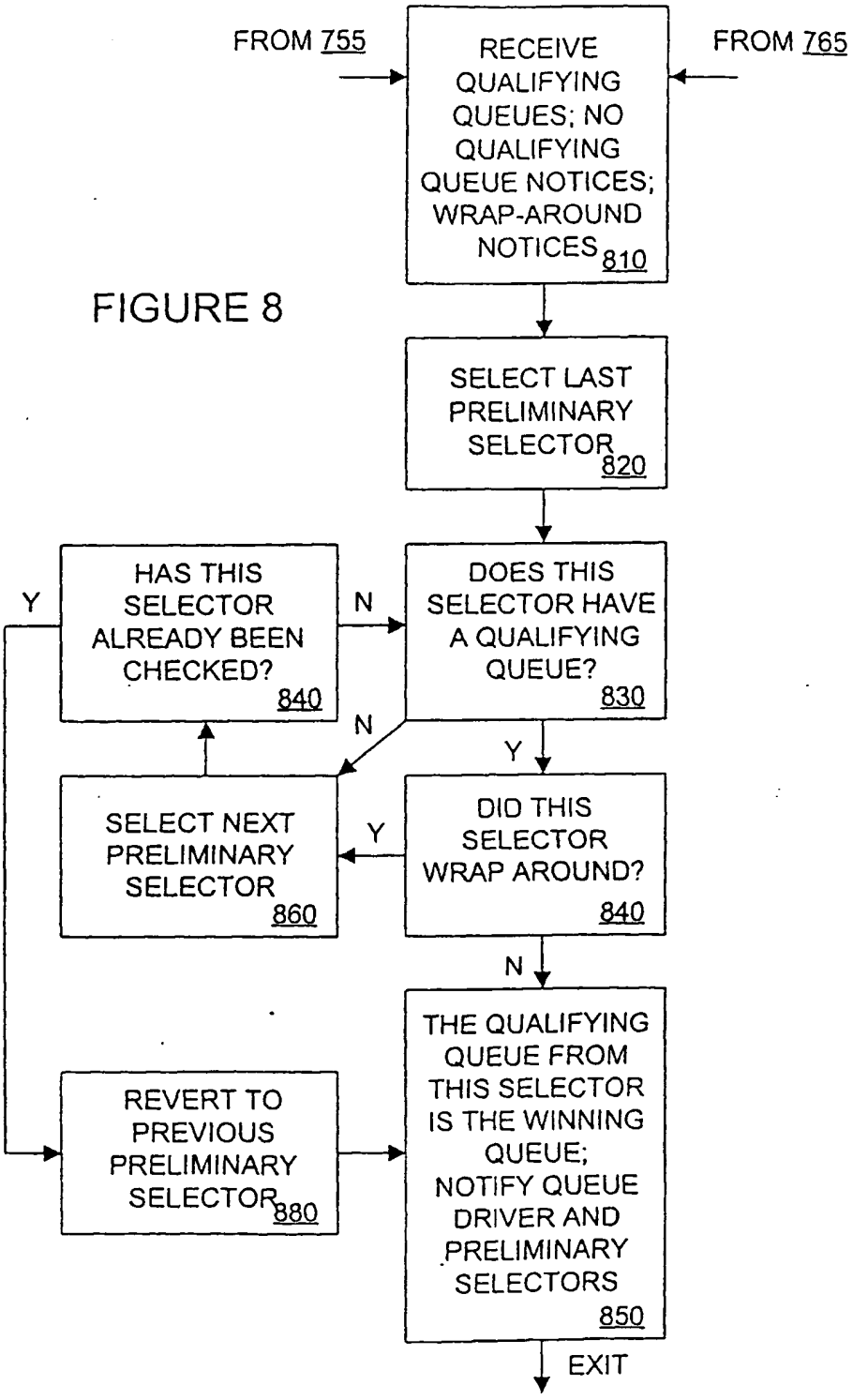

RECEIVE MY QUEUE INFO 710
FROM 650
WAS LAST WINNING QUEUE MY QUALIFYING QUEUE? 720
N
Y
SELECT MY NEXT QUEUE 730
SELECT MY FIRST QUEUE 740
HAS THIS QUEUE ALREADY BEEN CHECKED? 750
Y
NO QUALIFYING QUEUE; NOTIFY FINAL SELECTOR 755
GOTO 810
N
IS THIS QUEUE A PARTICIPATING QUEUE? 760
Y
THIS IS MY QUALIFYING QUEUE; NOTIFY FINAL SELECTOR 765
GOTO 810
N
IS THIS QUEUE MY LAST QUEUE? 770
Y
WRAP-AROUND; NOTIFY FINAL SELECTOR 780

FIGURE 8


FROM 755
FROM 765
RECEIVE QUALIFYING QUEUES; NO QUALIFYING QUEUE NOTICES; WRAP-AROUND NOTICES 810
SELECT LAST PRELIMINARY SELECTOR 820
DOES THIS SELECTOR HAVE A QUALIFYING QUEUE? 830
HAS THIS SELECTOR ALREADY BEEN CHECKED? 840
DID THIS SELECTOR WRAP AROUND? 840
SELECT NEXT PRELIMINARY SELECTOR 860
REVERT TO PREVIOUS PRELIMINARY SELECTOR 880
THE QUALIFYING QUEUE FROM THIS SELECTOR IS THE WINNING QUEUE; NOTIFY QUEUE DRIVER AND PRELIMINARY SELECTORS 850
EXIT
Y
N
N
Y
Y
N